IN THE CLAIMS:

Claims 1-10 remain pending in the present application and new claims 11-19 are added as follows:

1. (amended) A method for processing tasks in a data processing system including a microprocessor and an instruction cache wherein tasks of different types are defined in the system, each task type having code associated therewith, the tasks being processed in order by loading the associated code into the instruction cache for execution on the microprocessor, the method comprising the step of

placing the tasks of ~~at least one~~ the same task type into a batch such that the tasks in a batch are processed before processing the next ordered task.

2. (original) A method as claimed in claim 1 wherein the code associated with at least one type of task fits within the instruction cache, the method comprises the further steps of: processing such a task by loading the associated code into the instruction cache and executing the code on the microprocessor, and, on a determination that there is a further task of like type in the batch, executing the loaded code to process the further task.

3. (original) A method as claimed in claim 1 wherein the code associated with at least one type of task is not capable of being loaded as a whole into the instruction cache, the code being logically divided at one or more break points into two or more portions, and wherein during processing of such a task, the method comprises the further steps of responding to a break point defined within a first portion of the code to schedule a further task for future execution of a second portion of the code.

4. (original) A method as claimed in claim 3 wherein the further scheduled task is placed in a batch of like tasks.

5. (original) A method as claimed in claim 3 wherein each of portions of code defines an atomic operation.

6. (original) A method as claimed in any preceding claim, wherein a task is placed in a batch at the time the task is scheduled.

7. (original) A method as claimed in any preceding claim wherein the tasks are managed as a queue.

8. (amended) A computer program product comprising a computer usable medium having computer readable program code means embodied in the medium for processing tasks in a data processing system, the data processing system including a microprocessor and an instruction cache and wherein tasks of different types are defined in the system, each task type having code associated therewith, the tasks being processed in order by executing the associated code on the microprocessor, the program code means comprising code means for scheduling tasks of ~~like~~ the same type into a batch such that tasks in a batch are processed before processing the next ordered task.

9. (amended) Data processing apparatus comprising a microprocessor and an instruction cache wherein tasks of different types are defined in the system, each task type having code associated therewith, the apparatus including:

means for processing the tasks in order by loading the associated code into the instruction cache for execution on the microprocessor; and

means for scheduling tasks of ~~like~~ the same type into a batch, wherein the means for processing the tasks is operable to process the tasks in a batch before processing the next ordered task.

10. (original) Data processing apparatus as claimed in claim 9 wherein the microprocessor and i-cache are embodied on a single chip.

11. (new) A method for scheduling tasks in a task queue, the method comprising:

identifying a new task to be scheduled in the task queue;

determining if the task queue includes a cached task that requires substantially the same code to process the cached task and the new task; and

batching the new task with the cached task if the task queue

includes the cached task that requires substantially the same code to process the cached task and the new task.

12. (new) The method of claim 11, further comprising adding the new task to the end of the queue if the task queue does not include the cached task that requires substantially the same code to process the cached task as the new task.

13. (new) The method of claim 11, further comprising:

loading task code for processing the cached task into an instruction cache;

executing the task code for processing the cached task in the instruction cache; and

executing the task code for processing the new task in the instruction cache without loading new code into the instruction cache.

14. (new) The method of claim 13, further comprising:

determining if the task code is capable of fully loading into the instruction cache; and

if the task code is not capable of fully loading into the instruction cache, logically dividing the task code such that at least one substantially atomic portion of the task code will fully load in the instruction cache.

15. (new) A computer program product embodied in a tangible media comprising:

computer readable program codes coupled to the tangible media for scheduling tasks in a task queue, the computer readable program codes configured to cause the program to:

identify a new task to be scheduled in the task queue;

determine if the task queue includes a cached task that requires substantially the same code to process the cached task and the new task; and

batch the new task with the cached task if the task queue includes the cached task that requires substantially the same code to process the cached task and the new task.

16. (new) The computer program product of claim 15, wherein the program code is further configured to cause the program to add the new task to the end of the queue if the task queue does not include the cached task that requires substantially the same code to process the cached task as the new task.

17. (new) The computer program product of claim 15, wherein the program code is further configured to cause the program to:

load task code for processing the cached task into an instruction cache;

execute the task code for processing the cached task in the instruction cache; and

execute the task code for processing the new task in the instruction cache without loading new code into the instruction cache.

18. (new) The computer program product of claim 17, wherein the program code is further configured to cause the program to:

determine if the task code is capable of fully loading into the instruction cache;

if the task code is not capable of fully loading into the instruction cache, logically divide the task code such that at least one substantially atomic portion of the task code will fully load in the instruction cache.